

# Towards Robustness Analysis of Component-based Systems built on Imperfect Information

Claudia López  
Universidad Técnica Federico  
Santa María,  
Departamento de Informática,  
Avenida España 1680,  
Valparaíso, Chile  
clopez @ inf.utfsm.cl

Hernán Astudillo  
Universidad Técnica Federico  
Santa María,  
Departamento de Informática,  
Avenida España 1680,  
Valparaíso, Chile  
hernan @ inf.utfsm.cl

Javier Pereira  
Universidad Diego Portales,  
Escuela de Ingeniería  
Informática,  
Av. Ejército 441, Santiago,  
Chile  
javier.pereira @ udp.cl

## ABSTRACT

Component selection is a key step to build component-based software systems, but in practice available components' information is usually imprecise, incomplete, uncertain and/or unreliable; good components may pass unnoticed. This article presents the Azimut approach to generate alternate assemblies of Off-the-Shelf Component as candidate solutions at early project stages. Azimut deals explicitly with imperfect information, by modeling information on components as (score, credibility) pairs where credibility is assessed by evaluators themselves, and fitness of assemblies as solution candidates is evaluated with aggregated scores. This separation of evaluation and credibility enables robustness analysis of a solution set regarding the quality of available information, i.e. determining how sensitive is received architectural advice to changes in information. Thus, architects can explicitly tune their willingness to use imperfect information, possibly deciding to use only assemblies with high robustness, or allocating extra effort to improve some key components' information. A detailed example illustrates the approach and its use to support early architectural decisions.

## Keywords

software architecture, component selection, imperfect information, COTS, uncertainty

## 1. INTRODUCTION

Architects use their experience and knowledge to evaluate and compare alternate solutions, but they are hampered by imperfect knowledge of available software artifacts. The available OTS (Off-the-Shelf) components space is huge and highly changing, therefore architectural knowledge about components characteristics is usually imprecise, incomplete, uncertain and/or unreliable. Thus, OTS component selection requires to make architectural decisions in a risky context in which architects have to select the more effective

techniques to mitigate risks in early stages.

Imperfect information can be dealt with using specific mathematical formalisms. Only Cooper et al [7], Canfora and Troiano [5] and Gashi and Popov [9] approaches deal explicitly with uncertainty using fuzzy values and aggregation operators, or Bayesian approach to support component selection. However, these proposals don't support solution generation, but focus on evaluation.

In this paper we explain how the Azimut approach [15, 3, 4] supports robustness analysis of alternative solutions sets, and allows architects to make decisions according to their willingness to use imperfect information.

The reminder of this article is structured as follows: Section 2 characterizes the problem; Section 3 surveys some related work; Section 4 presents the Azimut approach and its dealing of imperfect information; 5 introduces the robustness analysis regarding information quality; Section 6 illustrates robustness analysis with an example; and Section 7 presents future work and conclusions.

## 2. MOTIVATION

Architects must evaluate and compare alternate solutions at early stages. When solutions must be built from available software components [22], some additional complexity sources arise:

- The available components space is huge and highly changing, and therefore architects' knowledge of components characteristics is likely to be imprecise, incomplete, uncertain and/or unreliable.
- Empirical testing and comparison of alternative components regarding achievable requirements frequently is economically impractical, and architects need to analyze and compare alternative component-based solutions in early phases using their prior knowledge about components.

Imperfect information can be dealt with using specific mathematical formalisms, such as fuzzy values for imprecision and rough sets for incompleteness. Since architects wanting to compare candidate architectures need to use whatever information is available. They need to compare and rank the non-arithmetic values used to characterize their components, and know the risk level related to component information's quality.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WOODSTOCK '97 El Paso, Texas USA

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

We are concerned with the problem of identifying candidate component assemblies from imperfect information about huge numbers of components. In this context, “OTS components” is to be taken as a coarse-grained COTS component.

### 3. COMPONENT SELECTION AND UNCERTAINTY

Systematic component selection methods [17, 2, 18, 20, 13, 8, 1, 6] that define key attributes to describe components, and use this descriptions to evaluate and compare alternate solutions, but they are not deal explicitly with the risks from using imperfect information as input of evaluation process.

Uncertainty in software design or architecture has been dealt in few articles [9, 10, 5, 11, 14, 7]. Only Cooper et al [7], Gashi and Popov [9], and Canfora and Troiano [5] have proposed techniques to deal explicitly with uncertainty. Cooper et al. [7] use fuzzy values to describe qualitatively non-functional requirements, and the fuzzy component specifications are ranked using a set of test queries. Gashi and Popov’s approach [9] uses the Bayesian approach to model a single attribute of a single COTS component, and the technique can be used for selection of an optimal pair of components when there is one attribute of interest (failure on demand in [9]). Canfora and Troiano’s technique [5] uses fuzzy values to describe uncertainty of basic judgements about quality requirements, and aggregation operators to produce overall assessment in a evaluation and selection process (of one component in [5]). None of these approaches support the alternative solution generation from uncertainty information; and multi-criteria decision making algorithms have only been dealt by Canfora and Troiano’s technique [5].

### 4. SOLUTIONS GENERATION FROM IMPERFECT INFORMATION

The Azimut approach [15, 3, 4] supports architects in generating component assemblies for a given set of requirements by using intermediate abstract constructs as stepping stones in a derivation chain (see Fig. 1).

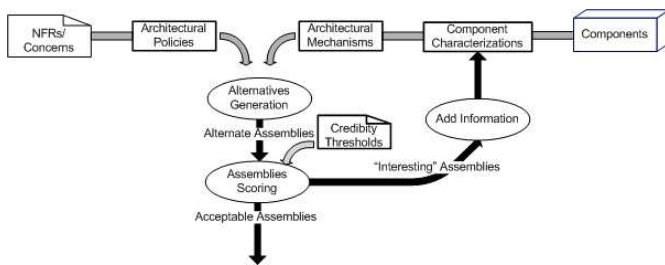


Figure 1: Matching process

Architects transform business problems to architectural problems reifying system quality requirements into categorized facets that describes architectural requirements kinds of each quality requirement domain (eg. persistent state replication for availability, or individual authorization for access control ) as shown in the left side of Fig.1. These categorized facets are named architectural policies in Azimut vocabulary.

Component evaluators maintain component descriptions specifying attributes of real components, but these descrip-

tions are constructed under imperfection conditions as evaluator’s personal judgements or incomplete knowledge about components space. Azimut uses component characterizations as component descriptions that include especial notation to describe the level kind and level of imperfect information about components.

Architects transform component characterizations to categorized facets that describes known design solutions for a specific quality requirement (eg. active replication for availability, or personal password for access control) that component implements. Azimut names architectural mechanisms at these design solutions. Components characterization and component-mechanism transformation are shown in the right side of Fig.1.

Then, architects search for solutions to architectural problem expressed as required architectural policies (left side) exploring the architectural mechanisms space and the components implementing them (right side). Hence, generation of alternatives is possible when architects match architectural policies with architectural mechanisms, and then identify components and components assemblies that implement all required architectural policies. Azimut maintains this architectural knowledge in catalogs and automates generation of alternatives. Since the information used to generate alternatives is itself imperfect, we provide operators and analysis to compare alternative candidates considering the information quality; this step is called ‘assemblies scoring’ in Fig. 1, and it allows architects to decide which alternatives are acceptable for a quality information thresholds, and which components need more information to be considered as part of acceptable solutions.

#### 4.1 Modeling Imperfect Information

Azimut catalogs [16] characterize artifacts using dimensions (categorized facets), but in practice this information may be partial and/or incomplete: since each dimension has several associated states (each category), *indeterminacy*, *imprecision* or *uncertainty* may appear if some stakeholders have trouble establishing which dimension state(s) do hold in a specific situation.

Each kind of imperfect information demands a particular representation and analytic approach:

- *Incomplete information* (indeterminacy) has been successfully dealt with by rough sets [?]; for example, our catalog(s) might not say anything about SMTP’s support for **Node Replication Consistency**.
- *Imprecision*; for example, SMTP may be said by some catalog author to “largely support” **active replication** since it is stateless and is not really hard to enhance or embed to operate in a replicated fashion.<sup>1</sup>
- *Uncertainty* (changeability of information) can also be subject to formal treatment, but we limit ourselves to let architects enact change by updating catalogs and propagating changes to second-order measures as required (and perhaps changing their decisions as well).
- *(Un)reliability* is usually modeled by probability statements; e.g. the above information may be tagged with a 90% reliability because we trust this catalog.<sup>2</sup>

<sup>1</sup>An architectural judgement, of course!

<sup>2</sup>We may actually trust the catalog’s author, but we sidestep complex trust issues in this article.

The latter is particularly relevant because architectural information gathered in catalogs are in general elaborated by different stakeholders, or even by third parties. This provides a way of sharing architectural information and of gathering complementary sources about any given artifact.

The credibility of scores may be assessed by the evaluator(s) themselves (as in a paper review process), by external assessors (e.g. perceived trustworthiness of the information source), or from a synthesis of such information (e.g. social networks-based [12]).

## 4.2 Alternative Assemblies Evaluation Using Imperfect Information

Consider an architectural scenario modeled as a set of desired properties or constraints to be satisfied, which an architect represents by a architectural policies set  $p = p_i | i = 1 \dots n$  where  $p_i$  is an specific architectural policy of a specific domain. For example, access control authorization policies are: individual, user groups and user roles authorizations.

Let  $\mu(x, y) \in \{1, 2, 3, 4\}$  be defined as the *credibility level* that an abstraction  $x$  supports an abstraction  $y$ ; then, a mechanism  $m \in M$  supports a policy  $p_j$  with credibility  $\mu(m, p_j)$ . Analogously, a component  $c \in C$  implements a mechanism  $m_{i,j}$  with credibility  $\mu(c, m_{i,j})$ . A component  $c_i \in C$  ( $i = 1, \dots, n_c$ ) is described with the set  $c_i = \{m_{i,1}, m_{i,2}, \dots, m_{i,n_m} \in M\}$  of mechanisms it implements. Intuitively, the scores count the arguments for a given component or assembly as solution to a given policy.

Given a  $n$ -item component assembly  $A$ , the **support score of a component assembly  $A$  for policy  $p$**  is

$$S_{assem}(p, A, \alpha, \beta) = \frac{|C(A, D_p)|}{|D_p|} \quad (1)$$

$|C(A, D_p)|$  is the number of policies in  $p$  satisfied by the components in  $A$ , and  $|D_p|$  is the total number of policies referenced by  $p$ , to be satisfied. Given that information on mechanisms and components may be unreliable [16], parameters  $\alpha$  and  $\beta$  denote the minimum credibility level for a mechanism-policy or component-mechanism relation, respectively.

$\alpha$  and  $\beta$  strongly determine artifacts complying a given policy. These parameters represent cut levels aiding architects to set how reliable the information available must be in order to accept a component or assembly. As can be seen below, architects may use these parameters to explore what artifacts are worth to be further analyzed. In which follows, we will consider that  $\alpha$  and  $\beta$  may change but  $\mu$  functions do not, meaning that evaluations of credibilities are given while the support score determination unfolds. Robustness analysis will be developed in order to explore the space of best candidate artifacts.

## 4.3 Generation algorithm

A generation algorithm to identify candidate component assemblies was introduced in [4], which allows imperfect information about the component-mechanism and mechanism-policy relations, and generates assemblies by joining components greedily. When the assembly's support score is greater or equal than a minimum expected score, it is selected as a candidate; otherwise, another component is joined to it and the combination is re-evaluated. The process continues until no new assemblies can be generated.

## 5. ROBUSTNESS ANALYSIS

*Robustness* is defined as the capability of a solution to do not change when external conditions change [19]. Flexible system are robust, which means that they adjust face to a modified environment. Therefore, the robustness analysis purpose is to identify conditions where a solution (i.e., a set of candidate assemblies) is stable [21]. We have established that the support score depend on the credibility thresholds. Altering the initial threshold values may conclude on new assemblies, which is not necessarily expected, or on remove assemblies from acceptable assemblies set. These kinds of situations are explored in this section.

Robustness analysis centered on thresholds  $\alpha$  and  $\beta$  is used to indicate architects how strong the arguments must be in order to consider that a given assembly supports a required policies set. In most problem domains,  $\alpha = 2(M)$  is a weak constraint and  $\alpha = 4(H)$  supposes a "crisp" problem, i.e. requiring certain, precise and complete information. In this section, we will explore how robust is a solutions set when the values of these parameters change.

Given that any argument for an assembly support is obtained both satisfying the mechanism-policy and the component-mechanism minimum credibilities, two situations are worth to study, namely, the impact of increasing/decreasing either  $\alpha$  or  $\beta$  upon the original solution.

*Definition 1.* The robustness of a solution set to a  $\Delta$ -change of parameter  $\gamma$  is the proportion  $k_\gamma(\Delta, p)$  of candidate assemblies not eliminated from the solution set when a change  $\Delta$  is applied to  $\gamma$ , for the policies set  $p$ .

Let us define  $\alpha^*(p)$  and  $\beta^*(p)$  as the *maximum* values that these parameters may have which do not change the original solution set for the policy  $p$ .

Similarly, let us define  $\alpha_*(p)$  and  $\beta_*(p)$  as the *minimum* values that these parameters may have which do not change the original solution set, i.e. no new candidates are generated.

*Definition 2.* The robustness range of a parameter  $\gamma$ , in relation to a policy  $p$ , is the interval  $I_\gamma(p) = [\gamma_*(p), \gamma^*(p)]$  where the parameter may take values in order that none candidate assembly is removed from the original solution.

## 6. EXAMPLE

Credibility scores are used to represent imprecise knowledge concerning components and mechanisms. Available components are linked to architectural mechanisms that they potentially implement, and these links represent imprecise and undeterminate "implementation" relationships. Analogously, relationships between mechanisms and policies are labeled with credibility scores. For simplicity, in this example we assume that mechanism-policy links have highest credibility; and only component-mechanism relationships are under imprecision conditions. Scores related to each imprecision level and indeterminacy of mechanism-component implementation links are shown in Table 1. These information is represented in mechanism catalogs and component catalogs (see Figure 2) [16].

As an example, consider the architectural policy  $p$  defining the following eight policies:

- *communication*: ( $p_1$ : synchrony = asynchronous;  $p_2$ : topology = 1:M;  $p_3$ : initiator = push;  $p_4$ : integrity over timeliness);

| Scores | Semantics                                    |
|--------|----------------------------------------------|
| A(4)   | Strongly implements, high credibility        |
| A(3)   | Strongly implements, medium-high credibility |
| A(2)   | Strongly implements, medium-low credibility  |
| A(1)   | Strongly implements, low credibility         |
|        | Unknown                                      |
| ⋮      | ⋮                                            |
| D(1)   | Strongly doesn't implement, low credibility  |

**Table 1: Example: credibility scores for support degree of Components for Architectural Mechanisms**

| COTS                  | Architectural Mechanisms |      |      |      |           |          |           |                    |                     |        |
|-----------------------|--------------------------|------|------|------|-----------|----------|-----------|--------------------|---------------------|--------|
|                       | SMTP                     | IM   | NNTP | RSS  | SMTP-Auth | POP-Auth | IMAP-Auth | Active Replication | Passive Replication | Voting |
| $c_1$ =SendMail       | A(4)                     | D(4) | D(4) | D(4) | A(4)      | D(4)     | D(4)      |                    |                     |        |
| $c_2$ =CounterMailSer | A(4)                     | D(4) | D(4) | D(4) | A(3)      | A(3)     | A(3)      |                    |                     |        |
| $c_3$ =DNews          | D(4)                     | D(4) | A(4) | D(4) | D(4)      |          | D(4)      |                    |                     |        |
| $c_4$ =LifeKeeper     | D(4)                     | D(4) | D(4) | D(4) | D(4)      | D(4)     | A(3)      | A(3)               |                     |        |
| $c_5$ =SurgeMail(Clu) | A(4)                     | D(4) | D(4) | D(4) | A(2)      | A(2)     | A(2)      | A(3)               | A(3)                |        |

**Figure 2: Partial Contents of Components Catalog**

- *security*: ( $p_5$ : authorization = individual;  $p_6$ : authentication = based on something the user knows);
- *availability*: ( $p_7$ : state replication = persistent;  $p_8$ : consistency = replicated-write).

For these policies, the generation algorithm propose SMTP or NNTP, SMTP-Auth and active replication as solution at architectural mechanism level. Table 2 shows the generated components assemblies (i.e. candidate solutions) and their credibilities for any expected policy, when  $\alpha = 2$  and  $\beta = 3$  (for details about generation, see [4, 3]). For each  $x : y$  entry in this table, “x” represents the mechanism-policy credibility and “y” the respective component-mechanism credibility for a state (or column).

If the minimum support score in the example is set to 0.7, the set of solutions is reduced from  $\|A\| = 255$  ( $\sum_{i=1}^5 \binom{5}{i}$ ) combinations to merely 4 candidate assemblies, namely rows 5, 8, 10, and 11. Even in absence of detailed technical knowledge about compatibility and collaboration among each assembly’s components, already some good candidates can be identified as a starting point for developing a fuller solution.

For instance, applying definition 1 on Table 2, we notice that  $k_\alpha(1, p) = 4/4$ , while  $k_\beta(1, p) = 0/4$ . This means that robustness of this solution set strongly depends on  $\beta$ , but it is not sensitive to  $\alpha$ ; hence, improving the quality of information about the available *components* might impact the chosen candidate solutions, but improving the quality of information about the *mechanisms* would not.

From definition 2, we have  $I_\alpha(p) = [0, 4]$  and  $I_\beta(p) = [3, 3]$ . This situation indicates architects that, if they want stronger arguments, they must procure new and better information to improve the credibilities of the component-mechanism relations. Also,  $I_\beta(p) = [3, 3]$  indicates that solution set would change if we modify the required credibility in component-mechanism relation: if it was 2, SendMail would be a new alternative solution; and if it is 4 all alternatives are eliminated. The problem of knowing which credibilities increase

**Table 2: Support score of component assemblies using  $\alpha = 2; \beta = 3$**

| row | policy $p$   | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ | $p_6$ | $p_7$ | $p_8$ | Support |
|-----|--------------|-------|-------|-------|-------|-------|-------|-------|-------|---------|
| 01  | $c_1$        | 4:4   | 4:4   | 4:4   | 4:4   | 4:2   | 4:2   |       |       | 4/8     |
| 02  | $c_2$        | 4:4   | 4:4   | 4:4   | 4:4   | 4:1   | 4:1   |       |       | 4/8     |
| 03  | $c_3$        | 4:4   | 4:4   | 4:4   | 4:4   | 4:1   | 4:1   |       |       | 4/8     |
| 04  | $c_4$        |       |       |       |       |       |       | 4:3   | 4:3   | 2/8     |
| 05  | $c_5$        | 4:6   | 4:3   | 4:3   | 4:3   | 4:3   | 4:3   | 4:3   | 4:3   | 8/8     |
| 06  | $(c_1, c_2)$ | 4:4   | 4:4   | 4:4   | 4:4   | 4:2   | 4:2   |       |       | 4/8     |
| 07  | $(c_1, c_3)$ | 4:4   | 4:4   | 4:4   | 4:4   | 4:2   | 4:2   |       |       | 4/8     |
| 08  | $(c_1, c_4)$ | 4:4   | 4:4   | 4:4   | 4:4   | 4:2   | 4:2   | 4:3   | 4:3   | 6/8     |
| 09  | $(c_2, c_3)$ | 4:4   | 4:4   | 4:4   | 4:4   | 4:1   | 4:1   |       |       | 4/8     |
| 10  | $(c_2, c_4)$ | 4:4   | 4:4   | 4:4   | 4:4   | 4:1   | 4:1   | 4:3   | 4:3   | 6/8     |
| 11  | $(c_3, c_4)$ | 4:4   | 4:4   | 4:4   | 4:4   |       |       | 4:3   | 4:3   | 6/8     |

is a combinatorial problem, matter of ongoing research by the Azimut team using an integer programming approach, and goes beyond the scope of this paper. Analogous analysis can be developed considering imprecision at mechanism-policy level.

## 7. CONCLUSIONS

This article has proposed a robustness analysis technique than can be executed using imperfect information about architectural entities, allowing comparisons and ranking among candidate architectures, and thus enabling architects to explore a design space according to their preferred credibility thresholds.

The Azimut approach makes use of multi-dimensional catalogs [16], which hold information on architectural *policies*, *mechanisms* and *components characterizations*; catalogs may themselves be elaborated by third parties or by stakeholders, and can grow incrementally from new data. The key aspect of the Azimut approach is the representation of quality attributes as architectural policies, their systematic reification into architectural mechanisms, and the latter’s reification into components. This approach has four key advantages:

- allows to generate components assemblies from quality requirements;
- allows to record imperfect information (incomplete, imprecise, uncertain and unreliable);
- allows to evaluate and compare whole component assemblies at once;
- allows to analyze robustness of proposed solutions according the preferred credibility thresholds.

Azimut relies in quantitative manipulation rather than the symbolic approaches taken by most proposals; we believe that this will allow dealing with much larger design spaces, but this conjecture has not been empirically tested as yet. The simple framework here introduced is currently being expanded to fully account for indeterminacy and imprecision, and ongoing work is developing a technique to generate component assemblies by using the evaluation measures without having to explore by hand the whole design space.

## 8. REFERENCES

- [1] C. Alves and J. Castro. CRE: A systematic method for COTS components selection. In *SBES 2001: 15th Brazilian Symposium on Software Engineering*, October 2001.
- [2] C. Alves and A. Finkelstein. Challenges in COTS decision-making: a goal-driven requirements engineering perspective. In *SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 789–794, New York, NY, USA, 2002. ACM Press.
- [3] H. Astudillo, J. Pereira, and C. López. Evaluating alternative COTS assemblies from unreliable information. In *QoSA 2006: Third International Workshop on Quality of Software Architecture*, volume LNCS 4214, 2006.
- [4] H. Astudillo, J. Pereira, and C. López. Identifying “interesting” component assemblies for NFRs using imperfect information. In *EWSA 2006: European Workshop on Software Architecture*, LNCS 2006.
- [5] G. Canfora and L. Troiano. The importance of dealing with uncertainty in the evaluation of software engineering methods and tools. In *SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 691–698, New York, NY, USA, 2002. ACM Press.
- [6] L. Chung and K. Cooper. COTS-aware requirements engineering and software architecting. *Software Engineering Research and Practice*, pages 57–63, 2004.
- [7] K. Cooper and L. Chung. Managing change in an OTS-aware requirements engineering approach. In *MPEC '05: Proceedings of the second international workshop on Models and processes for the evaluation of off-the-shelf components*, pages 1–4, New York, NY, USA, 2005. ACM Press.
- [8] K. Douglas and B. Laurence. Applying social-technical approach for COTS selection. In *Proceedings of 4th UKAIS Conference*. University of York, McGraw Hill, April 1999.
- [9] I. Gashi and P. Popov. Uncertainty explicit assessment of off-the-shelf software: Selection of an optimal diverse pair. *ICCBSS'07: Sixth International IEEE Conference on Commercial-off-the-Shelf (COTS)-Based Software Systems*, 0:93–102, 2007.
- [10] R. V. Giddings. Accommodating uncertainty in software design. *Commun. ACM*, 27(5):428–434, 1984.
- [11] I. Gorton and J. Haack. Architecting in the face of uncertainty: An experience report. In *ICSE '04: Proceedings of the 26th International Conference on Software Engineering*, pages 543–551, Washington, DC, USA, 2004. IEEE Computer Society.
- [12] P. Inostroza and H. Astudillo. Emergent architectural component characterization using semantic web technologies. In *SWESE 2006: Second Workshop on Semantic Web Enabled Software Engineering*, NOV 2006.
- [13] J. Kontio. A case study in applying a systematic method for COTS selection. In *ICSE '96: Proceedings of the 18th international conference on Software engineering*, pages 201–209, Washington, DC, USA, 1996. IEEE Computer Society.
- [14] P. A. Laplante and C. J. Neill. Uncertainty: A meta-property of software. In *SEW '05: Proceedings of the 29th Annual IEEE/NASA on Software Engineering Workshop*, pages 228–233, Washington, DC, USA, 2005. IEEE Computer Society.
- [15] C. López and H. Astudillo. Explicit architectural policies to satisfy NFRs using COTS. In J.-M. Bruel, editor, *Satellite Events at the MoDELS 2005 Conference: MoDELS 2005, Lecture Notes in Computer Science*, volume 3844, pages 227 – 236. Springer Berlin / Heidelberg, January 2006.
- [16] C. López and H. Astudillo. Multidimensional catalogs for systematic exploration of component-based design spaces. In *IWASE 2006: First International Workshop on Advanced Software Engineering. Proceedings of IFIP World Congress 2006*, August 2006. Springer.
- [17] C. Ncube and N. Maiden. PORE: Procurement-oriented requirements engineering method for the cbse development paradigm. In *International Workshop on Component-based Software Engineering*, May 1999.
- [18] M. Ochs, D. Pfahl, G. Chrobok-Diening, and B. Nothhelfer-Kolb. A COTS acquisition process: Definition and application experience. In *ESCOM '00: 11th European Software Control and Metric Conference*, 2000.
- [19] J. Pereira and B. Paulre. Flexibility in manufacturing systems: a relational and a dynamic approach. *Eur. J. Oper. Res.*, 130(1):70–85, 2001.
- [20] B. C. Phillips and S. M. Polen. Add decision analysis to your COTS selection process. *The Journal of Defense Software Engineering, Software Technology Support Center Crosstalk*, 2002 2002.
- [21] B. Roy. *Multicriteria Methodology for Decision Aiding*. Kluwer Academic Publishers, 1996.
- [22] C. Szyperski. *Component Software*. Addison-Wesley Professional, second edition, November 2002.